Create a Utility Host & Build an AWS ParallelCluster

If you want to set up your own Linux based AWS EC2 instance to run Land Data Assimilation (DA) v3 container, follow section1, "**Create a Utility Host**" section of this document. If you already have a Linux-based utility host to run the Land DA v3 container-based workflow, you can directly go to section 2, "**Build the Utilities: Packer and the Amazon Machine Image (AMI)"**.

A utility host is an AWS EC2 instance that configures services such as licensing, monitoring cluster nodes etc. It is created using Linux-based operating system such as Amazon Linux2 or Ubuntu. Uses can select "**t3.micro**" or **similar free tier eligible instance type** for the utility host. The utility host acts as an entry point to connect to AWS ParallelCluster environments.

Make sure the utility host is deployed in the same subnet that you plan to use for your AWS ParallelCluster so that network connectivity works correctly. The AWS parallel cluster will include a head node (identical to login node) as well as worker nodes (identical to compute nodes in traditional HPCs).

The Land DA v3 workflow can be run on AWS using AWS ParallelCluster. For that, users need to install the following utilities on the utility host:

- **HashiCorp Packer 1.15**: Opensource tool from HashiCorp to automate the creation of the amazon machine image
- **AWS ParallelCluster client version 3.13.2**: Opensource tool from AWS to create and manage HPC clusters in cloud

All build scripts, including scripts to build the Packer, Amazon cluster Image, and the Parallel Cluster are available in the **aws-landda-tutorial** repository.

All UFS-WM applications and components have been tested in the AWS region "**us-east-1 (N. Virginia)**". Functionality of these applications in other AWS regions is not guaranteed.

A **paid AWS account is required** to proceed with Land DA v3 container-based test run.

1. Create a Utility Host (i.e., an AWS instance)

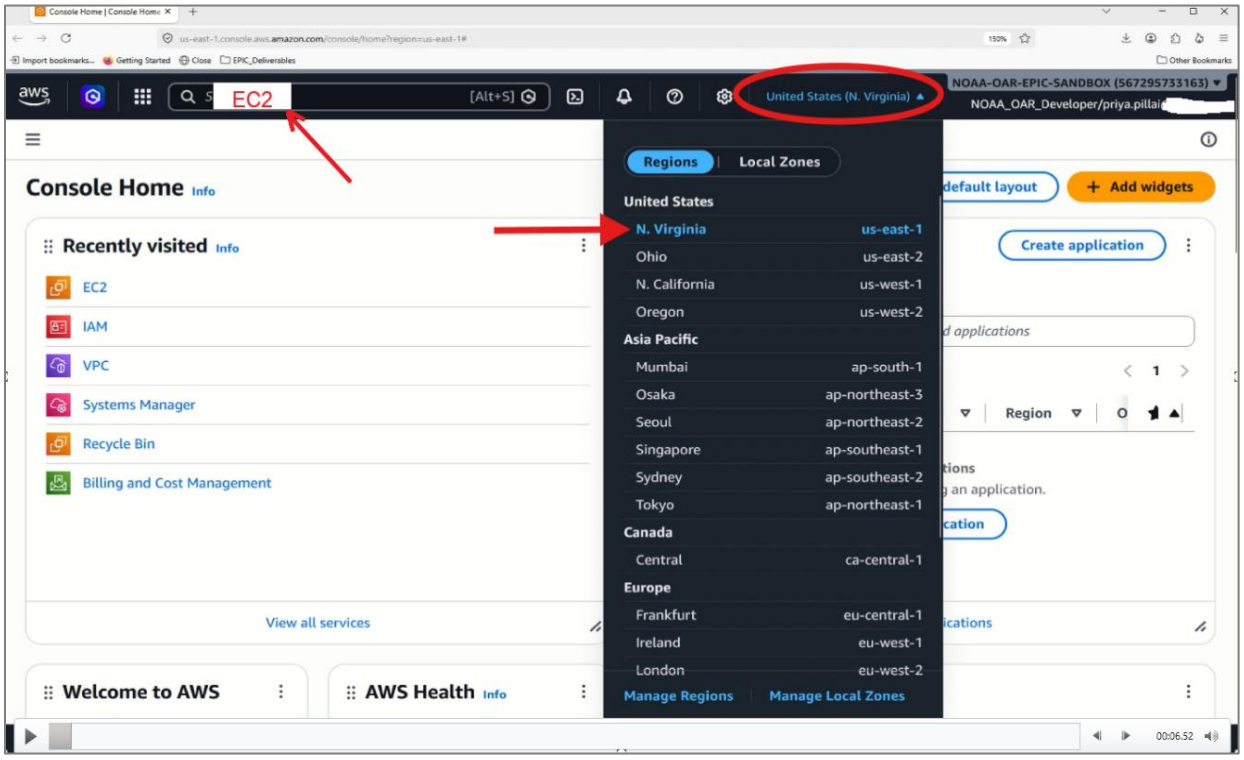
Login to AWS

(a) If you are using ***a personal AWS account***
You will need a paid AWS account to run Land DA experiments. Login at https://console.aws.amazon.com/, or sign up if you do not already have an account. This is ideal for individual learning & experimentation and is fully managed by the user. You should follow this "**Create a Utility Host and Build an AWS ParallelCluster**" guide. Once it is built, click on its "**instance-id**" to connect to the cluster.

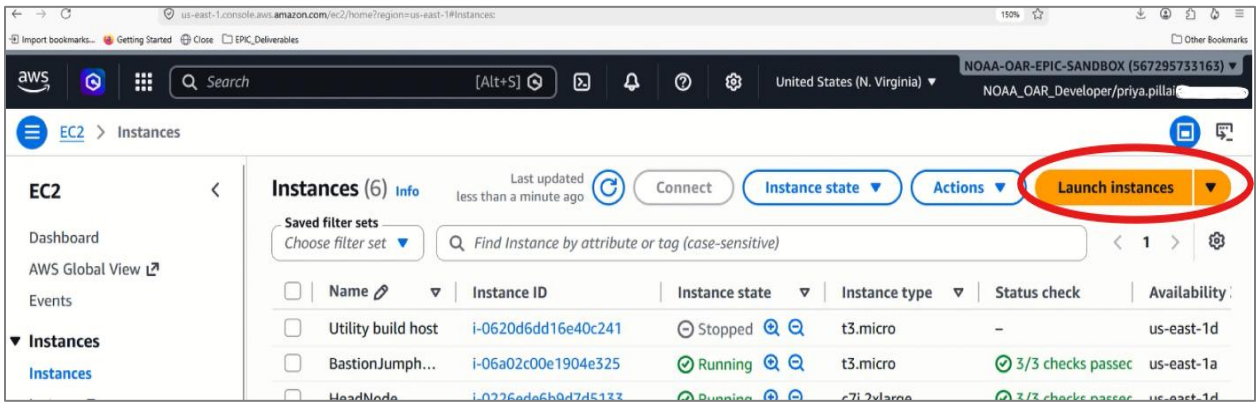
(b) If you are using an ***organization/institution provisioned AWS account (e.g. NOAA):***
Login through the **AWS SSO URL** shared by your cloud administrator. This account is governed by institutional policies, permissions, & security controls. You may need to coordinate with your cloud admin to build the cluster. Once the cluster is created, your cloud admin will share an "**instance-id**" for you to connect to it.

Navigate to AWS console and start creating an AWS EC2 instance

Once logged in, navigate to the AWS Console and select region to **United States (N. Virginia)**. Search for "**EC2**" in the search bar to open the **AWS EC2 service page**.

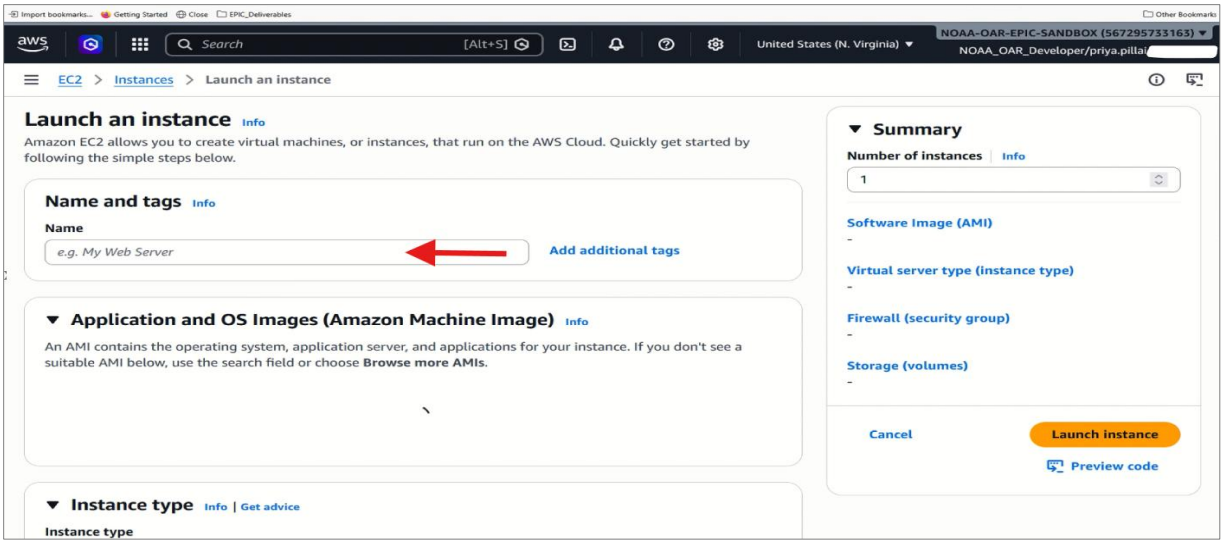


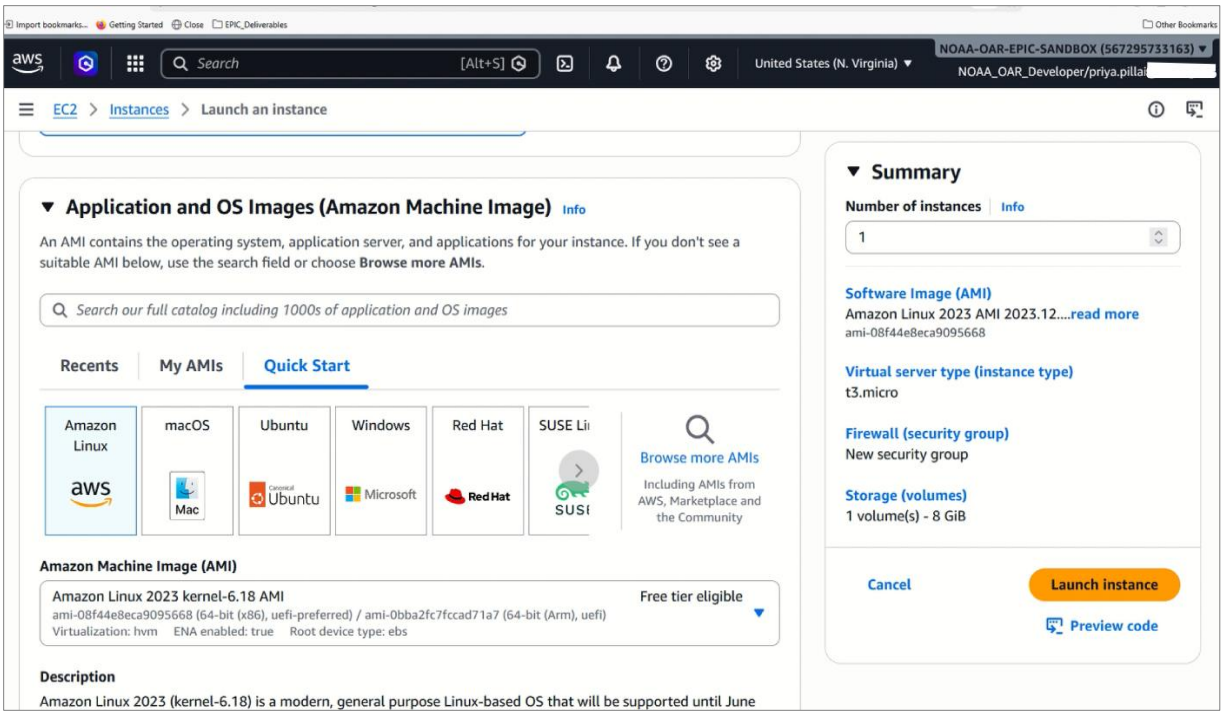
On this page, click the "**instances**" button to begin creating an AWS EC2 instance. Click on "**Launch instance"** button



Procced to fill in the "*Launch an instance"* page

- **Name and tags**: Create a name of your choice ("**`create_uhost-tutorial`**" is used in this documentation)
- **Application and OS Image:**
  - **AMI:** `Amazon Linux`
  - **Architecture:** `64-bit (x86)`
- **Instance type**: t3.micro



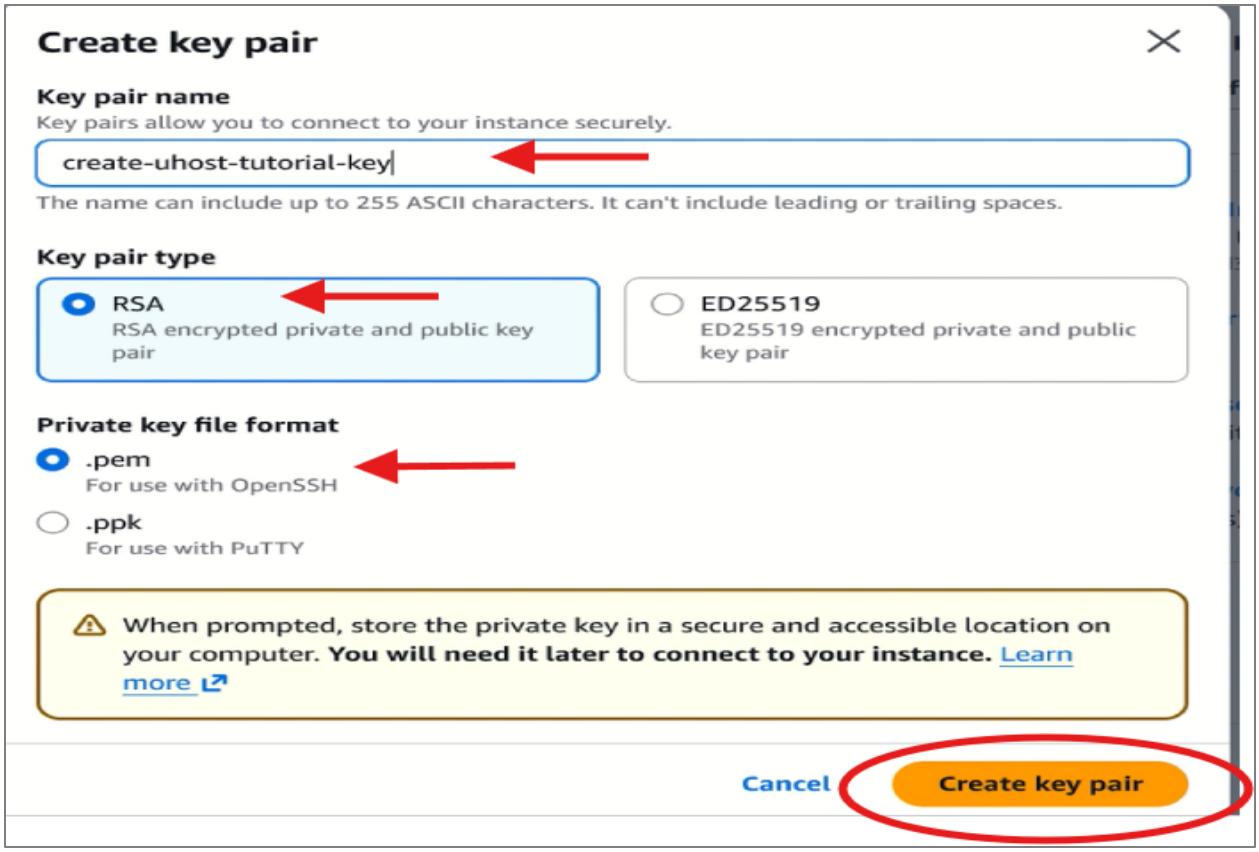


Important Note: ***Users can connect to AWS EC2 instances using either SSH or an AWS System Manager Session (SSM) Session Manager. SSH requires opening inbound port 22 in the instance's security group and using an SSH key pair.*** *For some organizations, security policies or firewalls do not allow port 22 to be opened, so users instead rely on the SSM Agent installed on the EC2 instance and AWS SSM Session Manager to establish a secure tunnel without exposing any inbound ports. Therefore, users need to choose between SSH and SSM Session Manager. In this tutorial, we use SSM Session Manager instead of traditional SSH.*

Create Key Pair for SSH connection

You need to set up an SSH keypair only if you plan to use SSH-based connection. You can **skip this step** if you already have a key pair and it shows up in the AWS Public key drop-down list. Click "**Create key pair**" and fill in or confirm the following fields. After creating the keypair, you must allow SSH traffic in the security group (port 22). No IAM role changes are needed for SSH.

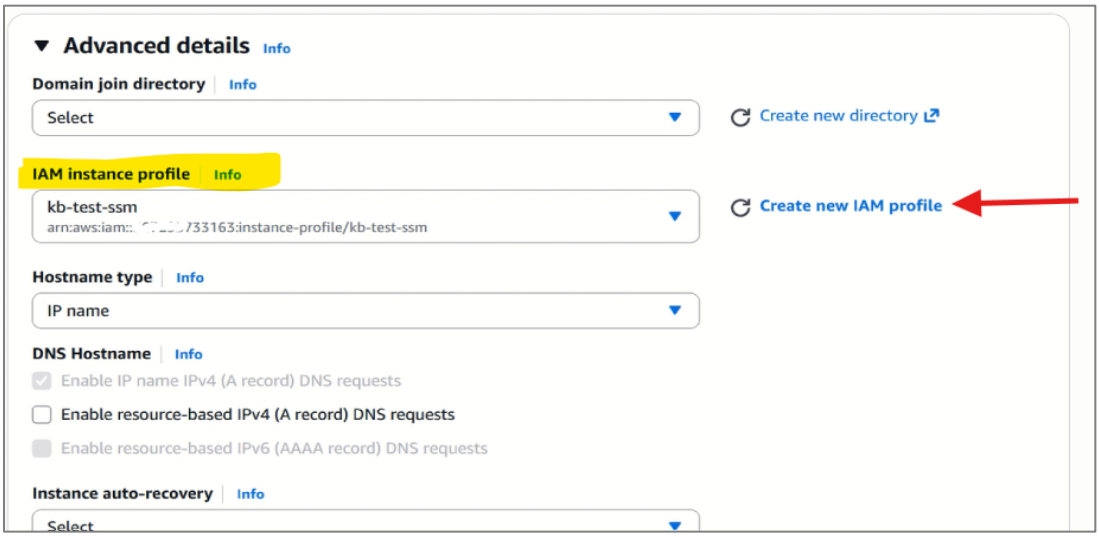
- **Key pair name:** `Create_UHost`
- **Key pair type:** `RSA` or `ED25519`
  You can choose any key pair type. The difference is in the encryption algorithm used.
- **Private key file format:** `.pem`
- Click **"Create key pair"**



This step **creates a key pair** consisting of a **public key** and a **private key**). AWS stores the public key, and you will be prompted to save the private key on your local machine. Make sure to remember where you save the private key. In the future, the public key will appear in the AWS drop-down list.

For SSM Session Manager-based connection:

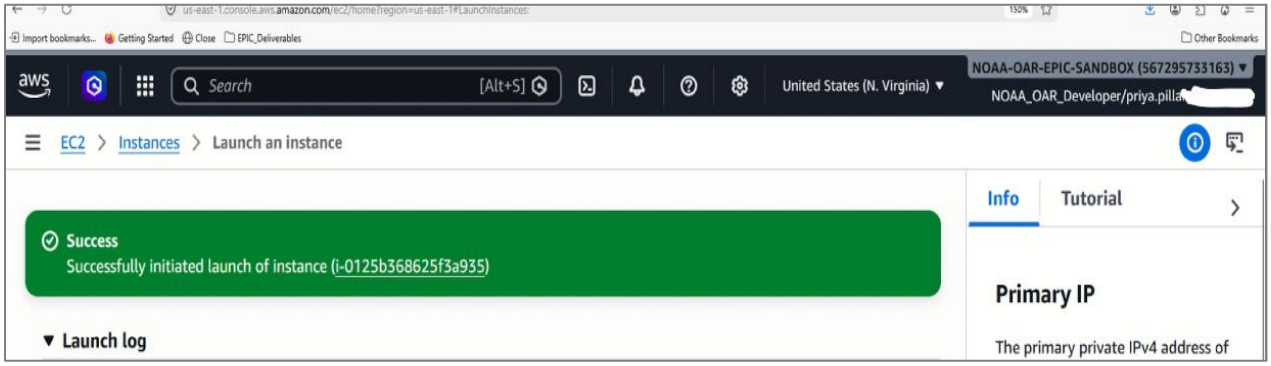
You do **not** need to select a key pair, and **SSH** should be disabled. You must attach an **IAM instance profile** that has SSM permissions. You may use an existing profile, or create a new IAM profile if one is not already available



- **Network settings**: `Leave as defaults`
- **Configure storage**
  - `1x 20 GiB gp3`
  - **File systems**: `None`
- **Advanced details**: `Leave as defaults`

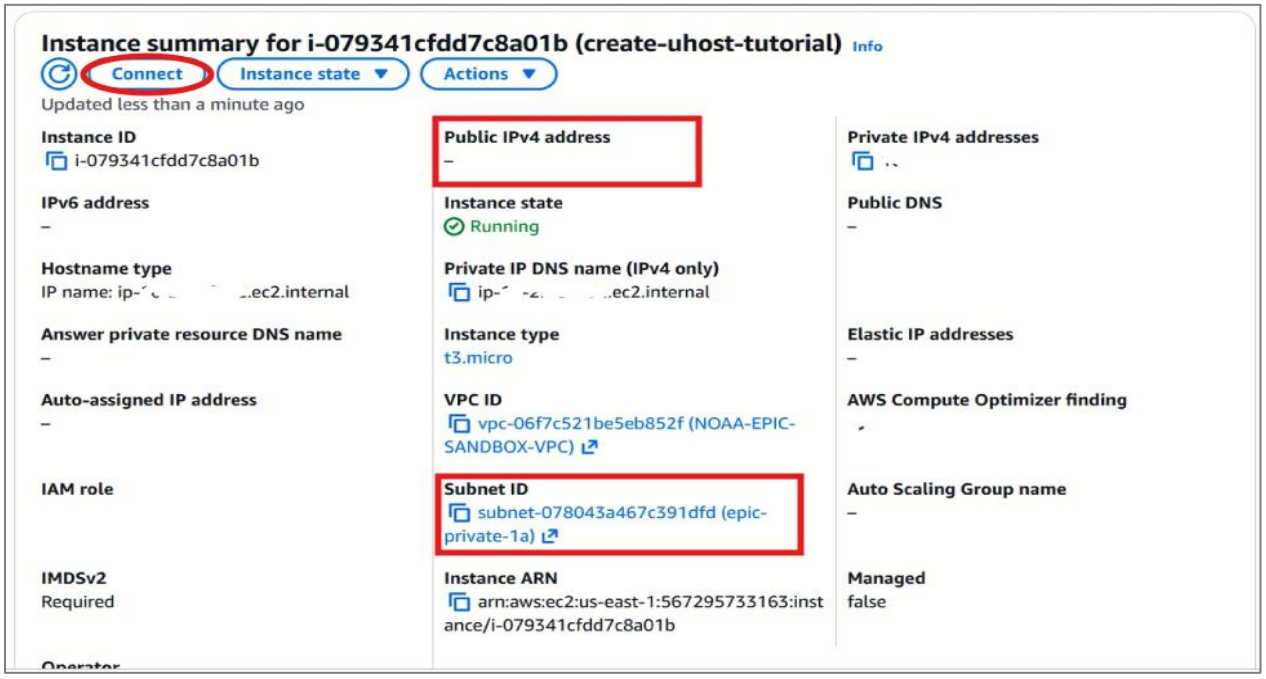
Launch the Utility host (i.e., AWS EC2 Linux instance) you just created

Once you have filled in all the required settings, click the "**Launch instance"** button on the right sidebar (or at the bottom, if using a narrow window) to start the **EC2 instance**. You will then see a success message displaying the **instance ID**, which links directly to the instance page.



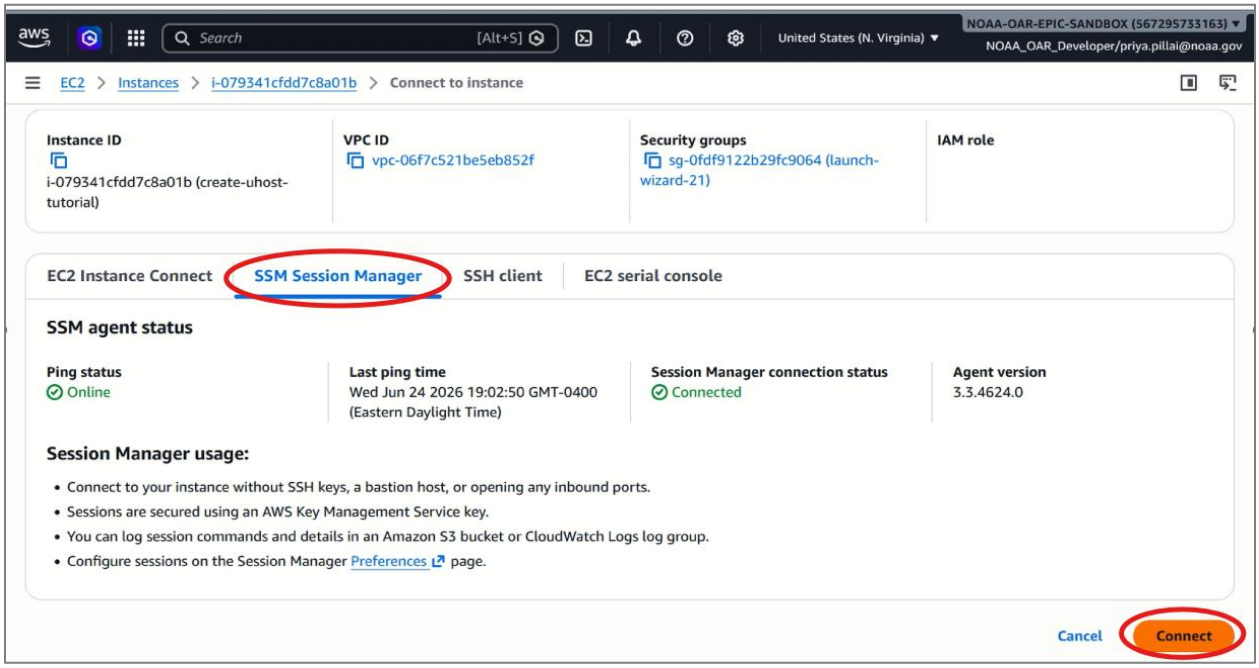
Click on the "**instance ID**" to open the instance page, which will display the **public IP** address for the instance. The example shown here uses a different "instance" than the one I created in the

"**Build a Utility host**" demo, as it was provided by my cloud admin. Make sure to note the **Public IPv4 address** and the **Subnet ID** of your instance, we will need them in the next step.



Connect to the Utility host using SSM Session manager or through SSH connection

Click the "Connect" button on the "**create-uhost-tutorial**" instance to connect to the utility host and proceed with building the cluster image.



2. Build Utilities and Amazon Machine Image (AMI)

From the **"create-uhost-tutorial"** instance, click on SSM Session Manager connection type and click "connect". From here users can proceed with building the Packer and with that packer, build the AWS cluster image (i.e, AMI) for Land DA. This AMI will contain the OS libraries, MPI, Python packages, and dependencies needed for Land DA. All build scripts, including scripts to build Packer, AMI, and ParallelCluster are shared in the **aws-landda-tutorial** repository.

Connect to Utility host. Users can use any Linux OS based host.

If your utility host is Amazon Linux 2 based

```
>> sudo su - ec2-user
```

If your utility host is Ubuntu based

```
>> sudo su - ubuntu
```

Copy "build_utility.sh" from **aws-landda-tutorial** GitHub Repository

Copy the packer build script, "**build-utility.sh**" from aws-landda-tutorial repo

```
>> wget https://github.com/NOAA-EPIC/aws-landda-tutorial/build-utility.sh
```

The next step is to run the packer build script, **build_utility.sh**

```
>> sudo ./build-utility.sh
```

Clone AWS ParallelCluster build scripts to build the Land DA v3 AWS cluster

```
>> git clone https://github.com/NOAA-EPIC/aws-landda-tutorial.git
```

```
>> cd aws-landda-tutorial
```

Use "screen" utility to run computationally intensive commands in the background

```
>> which screen
```

If you don't find screen installed, run

```
>> sudo yum install screen
```

And then run

```
>> screen
```

Set the environment variable AWS_REGION to **us-east-1**

```
>> export AWS_REGION=us-east-1
```

Disable colored output to avoid color codes clutter the terminal window

```
>> export PACKER_NO_COLOR=1
```

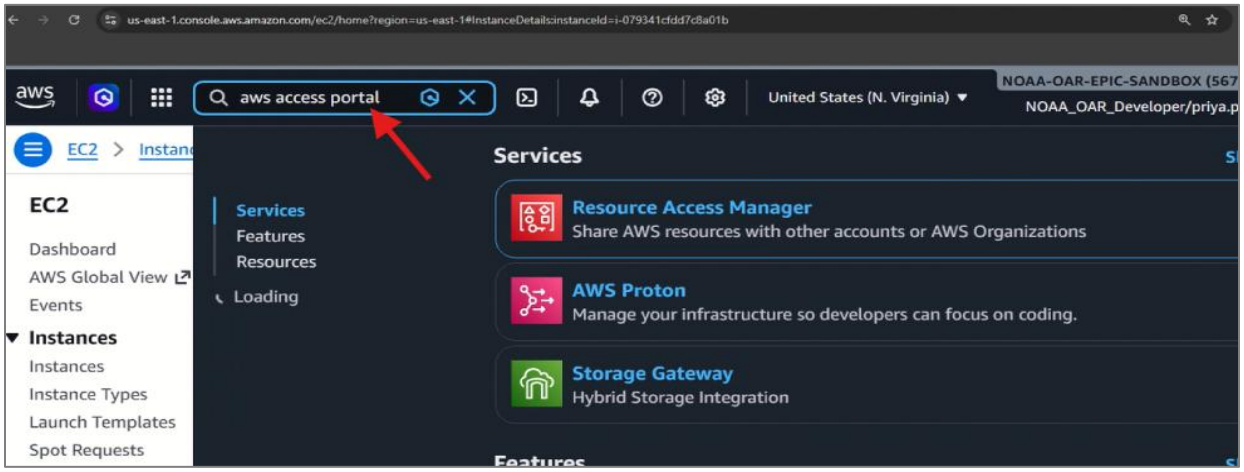
Enable **debug logging** for Packer and specify the file where Packer should write its logs

```
>> export PACKER_LOG=1
```

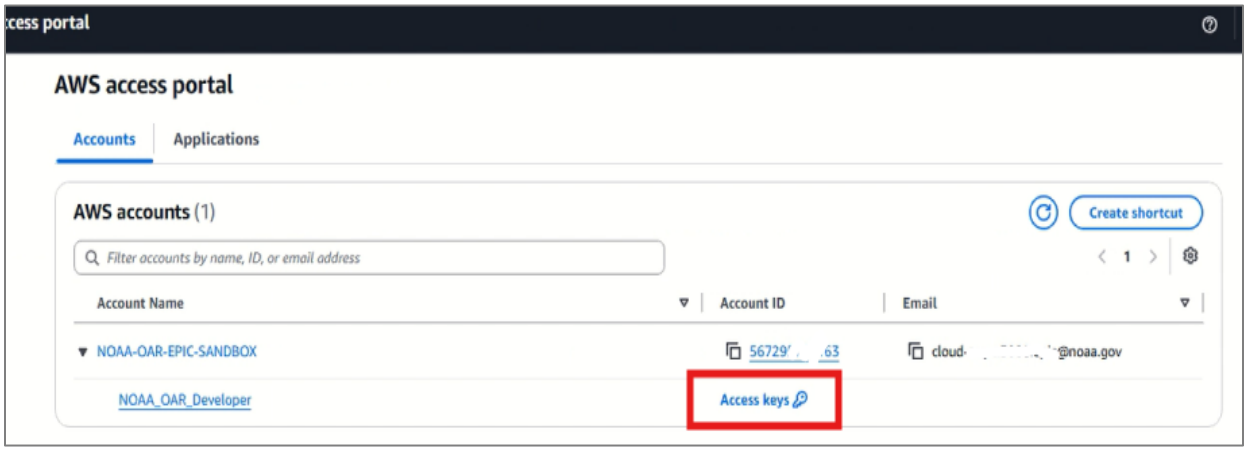
```
>> export PACKER_LOG_PATH="packer.log"
```

Copy AWS Environment variable

Go to AWS console, search for "**AWS Access Portal**"



Users will find a screen similar to the following and click on "**Access keys**"



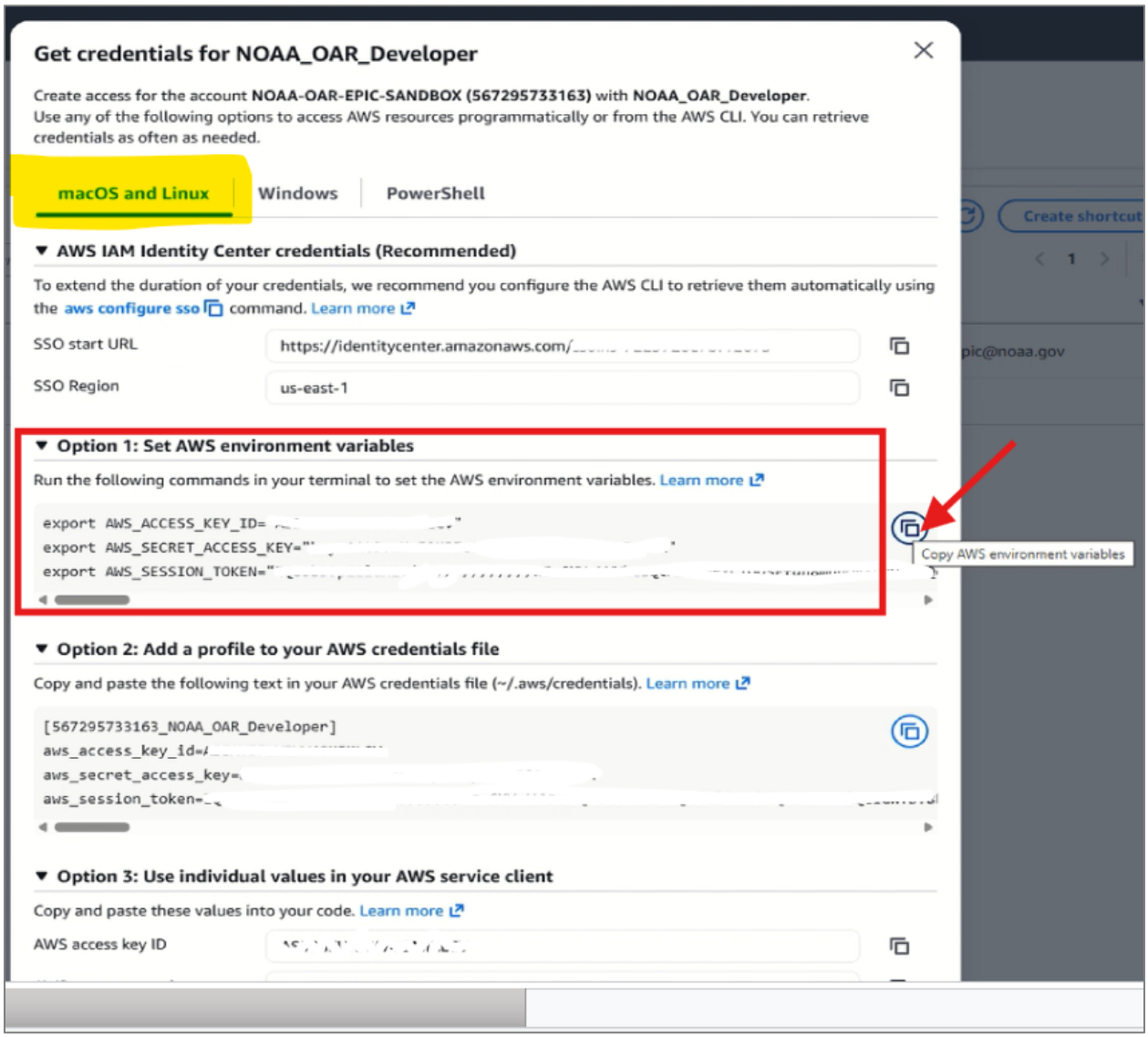
Copy the "**AWS environment variables**" using the copy button adjacent to the access keys on the access key page. These variables are:

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY &

AWS_SESSION_TOKEN

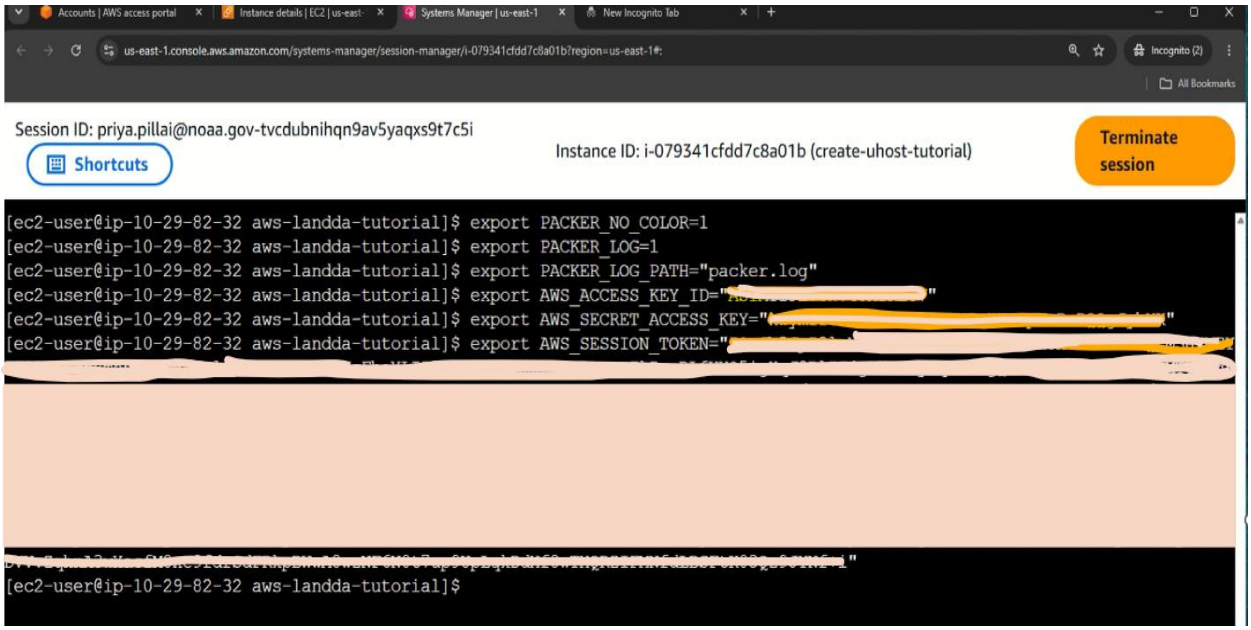
Please note that these variables are Sensitive and NEVER SHARE them with anybody.



Export the AWS Environment variables to the terminal

Paste the **AWS ACESS Key** credentials you copied to the terminal and it will automatically paste those variables one by one. Once again, these variables are Sensitive and NEVER SHARE them with anybody.

```
>> export AWS_ACCESS_KEY_ID = < your AWS variable>
>> export AWS_SECRET_ACCESS_KEY = < your AWS variable>
>> export AWS_SESSION_TOKEN = < your AWS variable>
```



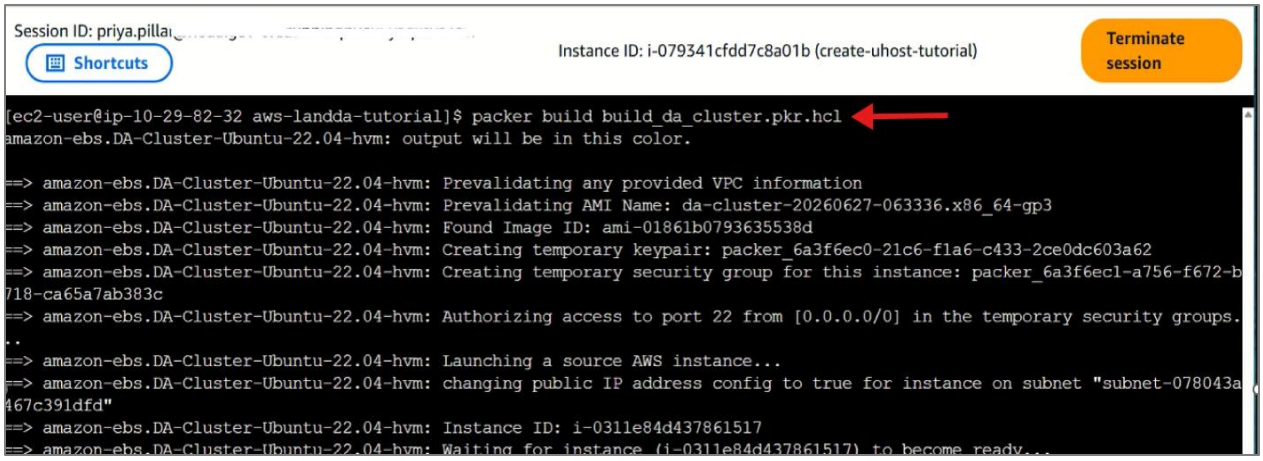
Build the packer and AWS Cluster Image for Land DA v3.0

Open "***build_da_cluster.pkr.hcl***" and update **subnet** field with the **subnet ID** of your utility host

```
>> vi build_da_cluster.pkr.hcl
```

You are now ready to build the initial baseline image. Run "***build_da_cluster.pkr.hcl***"

```
>> packer build build_da_cluster.pkr.hcl
```



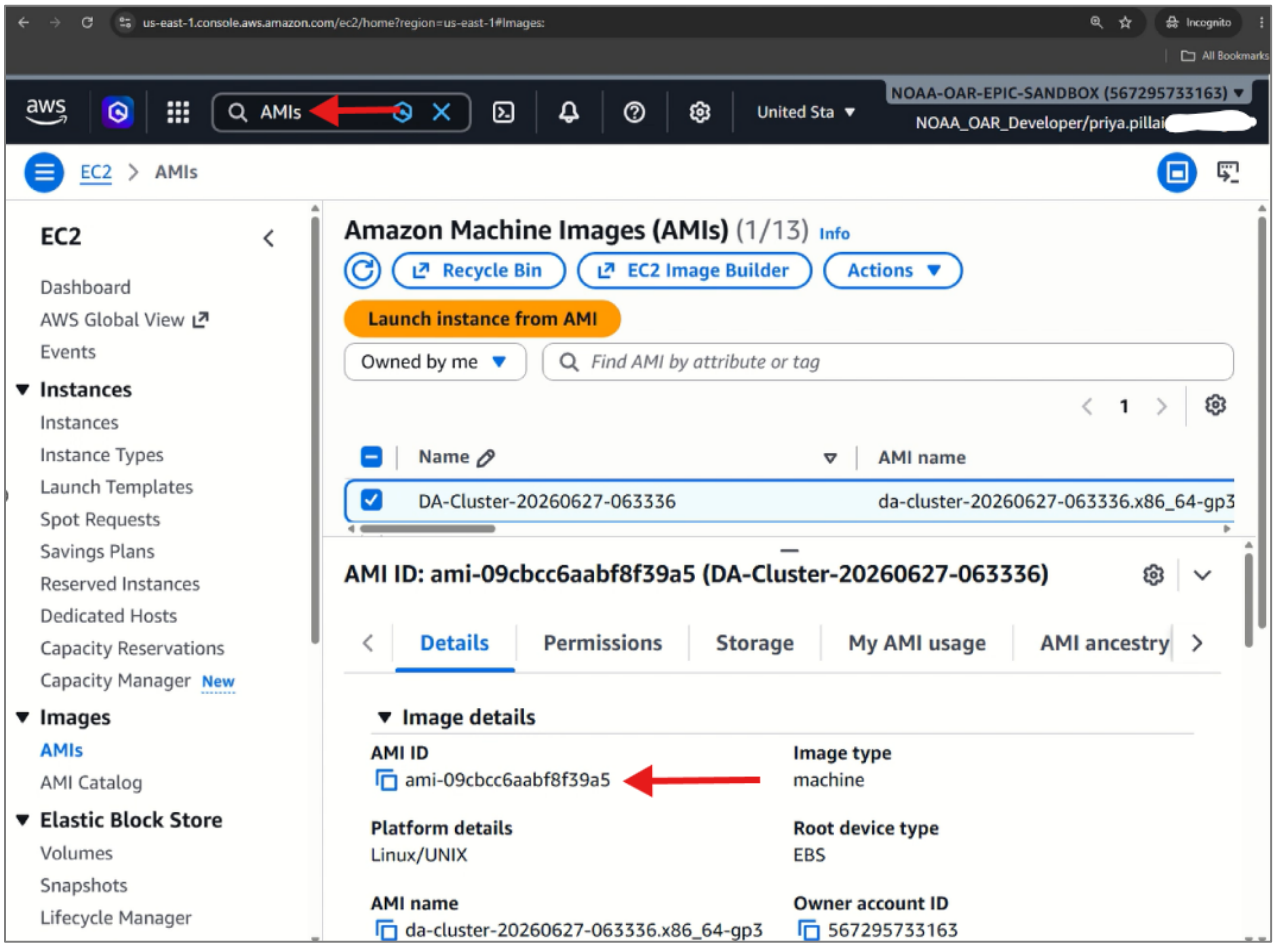
You can detach from the screen using the **Ctrl + (A followed by D)**, and the Packer build script will continue running in the background even after you terminate (i.e., close) the session. It takes **about 3 hours to build the cluster image**. Follow the progress through the **packer.log** file.

Once the build is complete, you will see this newly created AMI listed under **AWS AMIs**.

3. Spin up the AWS Parallel Cluster

Once the build is complete, you should see an AMI listed under your Amazon Machine Images. Take note of the **AMI ID**, as this will be used in the AWS ParallelCluster configuration.

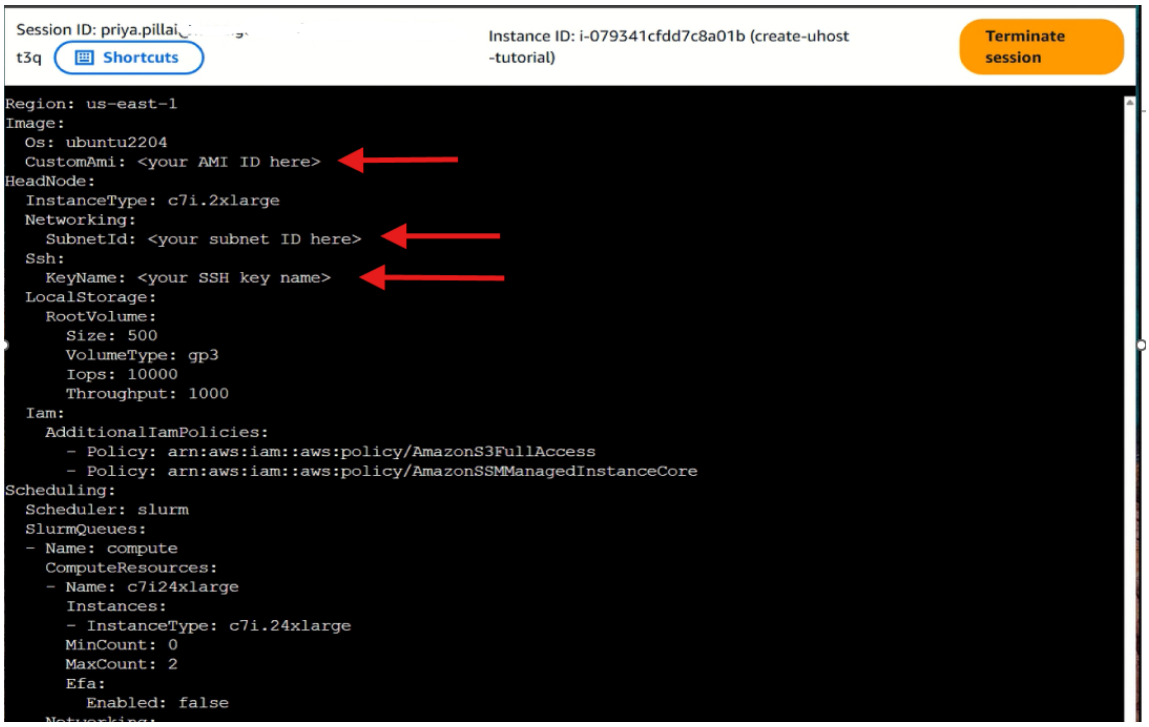
For AMI id, go to AWS console, search for AMIs, click on the AMI you created for this tutorial and copy the AMI ID.



Now, open the cluster build script, ***da_hpc.yaml*** and update the following to match your instances credentials. You already have the **Subnet ID** and (**Keypair name, if using SSH**). Now on the terminal,

>> `vi da_hpc.yaml`

- **`CustomAmi`**
- **`SubnetId`** (multiple places in this script)
- **`KeyName`**



You already have the **Subnet ID** and **Keypair** name. Now on the terminal, run the following command **to create the initial cluster**.

```
>> pcluster create-cluster –region us-east-1 –cluster-name landda-tutorial-cluster --cluster-configuration da_hpc.yaml
```

For additional information refer to the [AWS ParallelCluster documentation]

This process should take about **20 minutes** to complete. You can view the status of your cluster through the **AWS ParallelCluster CLI tool** by issuing the following command:

```
>> pcluster list-clusters --region us-east-1
```

The results should list the cluster status as **CREATE_COMPLETE** once the operation has successfully completed.